Z. Wu, H. Patel, M. Sachdev, and M. Tripunitara, "Strengthening PUFs using Composition," in proceedings of IEEE/ACM International Conference On Computer Aided Design (ICCAD), Nov. 2019, pp. 1–8. doi: 10.1109/ICCAD45719.2019.8942176.

# Strengthening PUFs using Composition

Zhuanhao Wu, Hiren Patel, Manoj Sachdev and Mahesh V. Tripunitara
University of Waterloo
Waterloo, Ontario, Canada
{zhuanhao.wu,hiren.patel,msachdev,tripunit}@uwaterloo.ca

## ABSTRACT

We explore the idea of composing PUFs with the intent that the resultant PUF is stronger than the constituent PUFs. Prior work has proposed a construction, which subsequent work has shown to be weak. We revisit this prior construction and observe that it is actually weaker than previously thought when the constituent PUFs are arbiter PUFs. This weakness is demonstrated via our adaptation of the previously proposed Logistic Regression (LR) attack. We then propose new constructions called PUFs-composed-with-PUFs ($P \circ P$). In particular, we retain a two-layer construction, but allow the same input to the composite PUF to be input to more than one constituent PUF at the first layer. We explore this family of constructions, with arbiter PUFs serving as the constituent PUFs. In particular, we identify several axes which we can vary, and empirically study the resilience of our constructions compared to the prior construction and one another from the standpoint of LR attacks. As insight in to why our family of constructions is stronger, we prove, under some idealized conditions, that the lower-bound on an attacker is indeed higher under our constructions than the upper-bound on an attacker for the prior construction. As such, our work suggests that composition can be a promising approach to strengthening PUFs, contrary to what prior work suggests.

## CCS CONCEPTS

• **Security and privacy → Embedded systems security**; **Hardware attacks and countermeasures**.

## KEYWORDS

Physical Unclonable Functions

**ACM Reference Format:**
Zhuanhao Wu, Hiren Patel, Manoj Sachdev and Mahesh V. Tripunitara. 2019. Strengthening PUFs using Composition. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

With the advent of Internet-of-Things (IoT) revolution, the number of distributed and unsupervised mobile computing devices continues to increase, and experts believe there will be approximately 100 billion connected devices by 2020 [16]. For such wide ranging devices, authentication for counterfeit prevention and secure communication is an important consideration. Physically Unclonable Functions (PUF)s have recently been proposed as replacements for non-volatile memories and on-die fuses that are prone to physical attacks for storing chip identifying digital signatures and seed generators to other cryptographic functions [9, 17]. PUFs use inherent manufacturing process variability to create circuits that appear physically identical at design time, but produce distinct, die-specific responses to input requests (or challenges) following fabrication. Each chip may contain many such challenge response-pairs (CRP)s.

PUF architectures can be categorized as strong or weak. The main difference is that a strong PUF must support a large CRP space. Over the years, several strong PUF architectures have been proposed [2]. However, most of these PUFs have also been shown to be susceptible to successful modeling attacks. Through modeling attacks, an adversary can mimic the behavior of the strong PUF with a high prediction accuracy (around 95% or higher) rendering them ineffective [1, 14]. An interesting approach proposed in [3] was to compose PUFs such that the resultant PUF offered improved security, which they called composite PUF (CPUF). The central thesis underlying the approach was that compositions allowed increasing the CRP space while also preserving the performance properties of the resultant PUF. In a later work, the authors themselves identified that the CPUF was also susceptible to a two-phase modeling attack [4] called the cryptanalysis attack (CA-ATK). They showed that CA-ATK, although successful in modeling CPUF, required an enumeration of a large CRP to conduct the attack.

In this paper, we start by reviewing CA-ATK, and show that for certain constructions of CPUF, its susceptibility to an attack is much worse than previously considered. Specifically, we propose an enhancement to CA-ATK that uses logistic regression (LR) called LR-CA-ATK to rid the need for the large number of enumerations on constructions of CPUF using arbiter PUFs (ARB-PUFs). Despite this, we contend that PUF compositions can offer an approach for strengthening PUFs even in the presence of the LR-CA-ATK, but, they require careful constructions. In particular, the manner in which the challenges are mapped to inputs of the PUFs can significantly contribute to the strengthening of the resultant composite PUF. We theoretically show that mapping functions can take a central role in strengthening the CPUF against the LR-CA-ATK, and also provide supporting empirical validation.

### 1.1 Contributions

We revisit the idea of composing PUFs, each of whose domains is smaller than $\{0,1\}^i$, to yield a PUF whose domain is $\{0,1\}^i$, and has strength $\Theta(i)$[1]. When we say "a strength of $\Theta(i)$," we mean that

[1]We use $\Theta(\cdot)$ to denote an asymptotic tight bound, and $O(\cdot)$ to denote an asymptotic upper-bound in a manner that is customary in computing[10].


*Conference'17, July 2017, Washington, DC, USA*

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Z. Wu, H. Patel, M. Sachdev, and M. Tripunitara, "Strengthening PUFs using Composition," in proceedings of IEEE/ACM International Conference On Computer Aided Design (ICCAD), Nov. 2019, pp. 1–8. doi: 10.1109/ICCAD45719.2019.8942176.



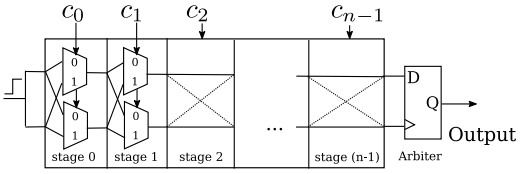



**Figure 1: The ARB-PUF architecture [2].**

the PUF is a realization of a random function $p\colon \{0,1\}^{f(i)} \to \{0,1\}$, where $f(i) = \Theta(i)$. Prior work suggests that composition is not an effective strategy to increase the strengths of PUFs; for example, the work of Sahoo et al. [4] yields a composition whose domain is $\{0,1\}^{nm}$, but is a PUF whose strength is $O(\max\{m,n\})$ only. We first propose a general model for compositions of PUFs of which the prior constructions are special cases. Two features of our model are that it allows an input to the composition to be input to more than one constituent PUF, and there are no restrictions on outputs of constituent PUFs serving as inputs to other constituent PUFs. We then propose restrictions on this general model so we can meaningfully study the improvement in strength relative to prior attempts at composition. We then prove, under some idealization assumptions, that a lower-bound for an attacker is strictly higher than the upper-bound on a prior construction. Indeed, our lower-bound on the attacker is $\Theta(i)$, for a composition whose domain is $\{0,1\}^i$. Thus, our analytical results suggest that even restricted forms of composition can be effective.

We carry out an extensive empirical assessment of our constructions. We first enhance the CA-ATK using logistic regression (LR-CA-ATK) that is known to be effective against prior constructions [3, 4]. We leverage insights from LR-CA-ATK to explore the use of mapping functions to strengthen composition of PUFs, and propose a family of PUF compositions $P \circ P$. Our work analytically establishes that mapping functions improve the resistance to attacks. We empirically evaluate several $P \circ P$ constructs against other state-of-the-art PUFs, and show its resilience. We also intuit properties that makes $P \circ P$ resilient to LR-CA-ATK.

## 2 RELATED WORK

Strong PUFs are used for authentication purposes while weak PUFs are often used for secret key generation where the CRP requirement is limited. As a security measure, each CRP is used once only. Consequently, when a PUF supports a large number of CRPs, i.e., is strong, adversaries cannot ascertain them under a constrained time frame [2]. Of course, the underlying function that the PUF realizes must be a random function, or some close approximation of it, for the PUF to be strong. Otherwise, even if the CRP space appears large, the PUF cannot be said to be strong, as it can be characterized fully with fewer CRPs than the size of its domain suggests.

Figure 1 shows the ARB-PUF [2], which is one of the most investigated strong PUFs. ARB-PUF has two identical delay paths that race from left to right through $n$ stages of multiplexers that are driven by the challenge bit. If the Data (D) arrives faster than the closing edge of the clock, the output (Y) makes a positive transition. An $n$ bit challenge directs these paths through $n$ multiplexers. The latch at the output acts as an arbiter selecting the edge arriving early [2]. The $n$ bit challenges result in $2^n$ unique path pairs, which is also known as the architecture's CRP space. There are several variants of the ARB-PUF that strengthen its security properties. For example, XOR-PUF [5] and LWS-PUF [7] both deploy multiple ARB-PUFs in parallel, but they use different strategies for processing challenge bits and producing response bits. Prior research demonstrated that realizations of strong ARB-PUF and its variants are susceptible to modeling attacks [1, 13, 14]. Amongst various ARB-PUFs, the LWS-PUF exhibits strongest resistance against machine learning modeling attacks [14].

Composite PUF [3] presented an approach to design strong PUFs using composition of PUFs. An important observation in their work was that PUF compositions increase the CRP space while preserving important performance properties [3]. Their work focused on a two-layer composition where the outputs of the first layer are fed as challenge inputs to the second layer PUF [4]. While their compositions were successful in increasing the CRP space, the proposers of this approach themselves identified a cryptanalysis attack (CA-ATK) that successfully modeled the composite PUF [4]. However, CA-ATK required an enumeration of the entire CRP space to be successful. In this work, we show that an enhanced version of the CA-ATK with machine learning could significantly improve the performance of the attack. Similarly, MPUF [8] is a composite PUF with two layers, which is susceptible to a modeling attack. Although recent research on PUF compositions has rendered the resulting architectures susceptible to attacks, we believe that the idea of composing PUFs is of genuine scientific value.

## 3 PUFS FROM COMPOSITION

We now introduce a general model for how PUFs can be composed to yield other PUFs. Then, we discuss the family of PUFs within that model on which we focus in this paper. That family includes, as special cases, certain constructions from prior work [3].

A PUF is a physical realization of a function, $p\colon \{0,1\}^i \to \{0,1\}$, i.e., it maps an $i$ bit input to a one bit output. Its intent is to serve as a random function, i.e., a function chosen uniformly from the set of all functions that map $i$ bits to one bit. We perceive a PUF that results from composition as a directed graph, $P = \langle V_P, E_P \rangle$; we show two examples in Figure 2. Each vertex $u \in V_P$ is a PUF. Each edge, $e \in E_P$, maps a PUF to an input of another PUF. Thus, $E_P$ can be perceived as a relation, $E_P \subseteq V_P \times V_P \times \mathbb{Z}^+$, where $\mathbb{Z}^+$ is the set of positive integers, where $\langle u, v, j \rangle \in E_P$ means that the output of the PUF $u$ is provided as the $j^{\text{th}}$ input of the PUF $v$. Our constraints are that given an edge $\langle u, v, j \rangle$, where $v\colon \{0,1\}^{i_v} \to \{0,1\}$, (i) $1 \leq j \leq i_v$, and, (ii) for every $\langle v, j \rangle$, there is exactly one edge incident on it, i.e., there is exactly one $u \in V_P$ such that $\langle u, v, j \rangle \in E_P$. In Figure 2, in the PUF to the left, the output of the constituent PUF $c_1$ is the first input to each of $c_2$ and $c_3$. So we have edges $\langle c_1, c_2, 1 \rangle$ and $\langle c_1, c_3, 1 \rangle$.

To represent inputs to and outputs from the composed PUF as a whole, we assume that we have two distinguished sets of edges, *inp*, *outp* $\subseteq E_P$, where each edge in *inp* has no source vertex, and each edge in *outp* has no destination PUF input. That is, the former is of the form $\langle \cdot, v, j \rangle$, and the latter is of the form $\langle u, \cdot, \cdot \rangle$. In Figure 2,

Z. Wu, H. Patel, M. Sachdev, and M. Tripunitara, "Strengthening PUFs using Composition," in proceedings of IEEE/ACM International Conference On Computer Aided Design (ICCAD), Nov. 2019, pp. 1–8. doi: 10.1109/ICCAD45719.2019.8942176.



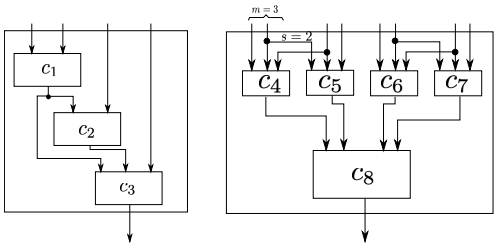



**Figure 2: Examples of PUFs from composition of other PUFs. To the left, the constituent PUFs are in three levels to yield a composition whose domain is $\{0,1\}^4$. To the right is a composition with domain $\{0,1\}^8$, with the constituent PUFs in two layers. The PUFs at the first layer, $c_4$ and $c_5$, are each 3-input PUFs, and share two inputs to the composition.**

the two inputs to the far left are input to the constituent PUF $c_1$. The output of the constituent PUF $c_3$ is the output of the composition.

*Restrictions.* We now consider restrictions to the above rather general model for composition. As Section 4 establishes, notwithstanding such restrictions, we can realize PUFs which, asymptotically, yield the maximum possible attack-resistance. Futhermore, such restrictions yield more feasible constructions in practice. We present increasing restrictions that culminate in the family that captures prior work on compositions, and on which we focus.

As a first restriction, we require that the graph $P = \langle V_P, E_P \rangle$ is acyclic. Both PUFs in Figure 2 are acyclic. By acyclic, we mean that the conventional directed graph $\langle V_P, F_P \rangle$, where $F_P = \{\langle u, v \rangle \in V_P^2 \mid \langle u, v, j \rangle \in E_P \text{ for some } j\}$, is acyclic. Given such a PUF that is acyclic, we can consider its topologically sorted version, i.e., one in which all edges go from left to right only. This allows us to associate a *level* with each constituent PUF. The PUF to the left in Figure 2 is a composition of three PUFs, each at a level. We could further constrain the levels to be the stricter *layers*. A constituent PUF is at Layer 1 if the only edges incident on it are those from *inp*, the set of inputs to the composition as a whole. Then, for a PUF at layer $l > 1$, all edges incident on its inputs are those from PUFs at layer $l - 1$. The PUF to the right in Figure 2 is such a layered construction.

*Family on which we focus.* The family of PUFs from composition on which we focus are layered constructions, with two layers only, Layers 1 and 2. The inputs to the composition are inputs to the constituent PUFs at Layer 1. The outputs from the PUFs at Layer 1 serve as inputs to a single PUF at Layer 2. The output from the constituent PUF at Layer 2 is the output to the composed PUF as a whole. An example is to the right in Figure 2.

More specifically, we associate our compositions with four parameters. (i) The number of inputs to the composition as a whole, $i$. In the PUF to the right in Figure 2, $i = 8$. (ii) The number of inputs to each constituent PUF at Layer 1, $m$. We adopt the restriction that each PUF at Layer 1 takes the same number of inputs. In the PUF to the right in Figure 2, $m = 3$. (iii) The number of *partitions*, $r$, on the inputs, where the inputs in a partition serve as input to only a subset of the Layer 1 constituent PUFs. Each such partition on the inputs then induces a partition on the Layer 1 PUFs, which are the ones that are fed those inputs. In the PUF to the right in Figure 2, we have $r = 2$ partitions. The first four inputs are in one partition, and the others are in the other. This partition on the inputs induces the partition $\{c_4, c_5\}, \{c_6, c_7\}$ on the Layer 1 PUFs. Last, (iv) the number of input bits that each PUF at Layer 1 has in common with another PUF at Layer 1, denoted $s$.

In the PUF to the right of Figure 2, $s = 2$, because each of $c_4, \ldots, c_7$ shares 2 bits of input with another PUF at Layer 1. We adopt the restriction that the $s$ is the same for all Layer 1 PUFs.

*Notation.* The notation we adopt to denote a PUF in our family is $[\mathbf{i}, \mathbf{m}, \mathbf{r}, \mathbf{s}]$ . For example, the PUF to the right in Figure 2 is denoted $[\mathbf{i} = 8, \mathbf{m} = 2, \mathbf{r} = 2, \mathbf{s} = 2]$.

## 3.1 The Cost of Realizing Compositions

We must acknowledge that we cannot get PUFs that are strong and resistant to attack without a cost. In Section 7, we characterize the hardware cost, in actual numbers, of our constructions. Here, we give a more abstract characterization.

The cost of constructing PUFs for the family of compositions we address can be characterized meaningfully as the size and number of constituent PUFs. The dominant factor is the number of Layer 1 PUFs. For $[\mathbf{i}, \mathbf{m}, \mathbf{r}, \mathbf{s}]$, the number of Layer 1 PUFs is $q = r\left(\left\lceil \frac{\frac{i}{r} - m}{m - s} \right\rceil + 1\right)$. We observe that if $s$ is large, e.g., $s = m - 1$, then the worst-case for $q$ is $\Theta(i)$; that is, the number of Layer 1 PUFs corresponds to the number of inputs. In this case, one may argue that a composition is not necessary at all, because if we have $\Theta(i)$ Layer 1 PUFs, then the Layer 2 PUF takes $\Theta(i)$ inputs. Therefore, we may as well simply not have the Layer 1 PUFs at all, and merely adopt the Layer 2 PUF as our PUF. However, we observe that for other values of $s$, the composition can perform quite well from the standpoint of cost. When $s = \Theta(1)$ and $r = \Theta(1)$, e.g., if $s = 1, r = 1$, then $q = O\left(\frac{i}{m}\right)$. We argue that this cost is quite favorable. For one thing, it is, asymptotically, the same cost as the CPUF. However, as we establish in the next section, we get better attack-resistance than the CPUF. For example, if $m \leq \frac{i}{2r}$, then the lower-bound for an attacker from Claim 2 in the next section exceeds the $\frac{i}{m} 2^m + 2^{\frac{i}{m}}$ upper-bound on an attacker for the CPUF.

In summary, security is not without cost. However, we observe that with the kinds of compositions our model admits, we can trade cost for better security.

# 4 ANALYSIS OF RESISTANCE TO ATTACK

We now establish, analytically, the level of resistance of an instance from our family of PUFs to attack. We assume an attacker that seeks to fully characterize a PUF. By that we mean the following. Given a PUF $p : \{0,1\}^i \rightarrow \{0,1\}$, we say that $p$ is fully characterized by a function $f : \{0,1\}^i \rightarrow \{0,1\}$ if $f(x) = p(x)$ for all $x \in \{0,1\}^i$ with probability 1. In practice, we typically relax this probability, e.g., to 95% only. We say that an attacker has fully characterized $p$ when he is in possession of such an $f$. This is the same attacker characterization as has been adopted in the literature.

An attacker is provided the following capabilities. (i) Black-box access to $p$. That is, the attacker is allowed to exercise $p$ with inputs, and observe the corresponding outputs. And, (ii) the attacker knows the design of the composition. That is, in our case, he knows $i, m, r$

Z. Wu, H. Patel, M. Sachdev, and M. Tripunitara, "Strengthening PUFs using Composition," in proceedings of IEEE/ACM International Conference On Computer Aided Design (ICCAD), Nov. 2019, pp. 1–8. doi: 10.1109/ICCAD45719.2019.8942176.



and $s$ given a PUF $[\mathbf{i}, \mathbf{m}, \mathbf{r}, \mathbf{s}]$, and the manner in which the PUF is constructed from those parameters. Thus, the only thing the attacker does not know are the actual functions that the constituent PUFs realize.

We quantify the strength of a PUF's resistance to attack as the number of queries the attacker needs to perform to the black-box. If for a PUF $p$, the attacker must perform $n_p$ queries to the black-box, and for another PUF $q$ the attacker must perform $n_q$ queries, and $n_q < n_p$, then we deem the PUF $p$ to be strictly more resistant to attack than the PUF $q$.

In our analysis below, we make the following idealization assumption. We assume that each constituent PUF is a random function. The reason we do this is that, our analysis pertains really to the manner in which we compose, rather than some artifact of the constituent PUFs.

*Notation.* Our focus is the Layer 1 PUFs in the composition. Given a composition $[\mathbf{i}, \mathbf{m}, \mathbf{r}, \mathbf{s}]$, as we point out in the previous section, the number of Layer 1 PUFs is $q = r\left(\left\lceil \frac{\frac{i}{r}-m}{m-s} \right\rceil + 1\right)$. We denote these as $c_0, c_1, \ldots, c_{q-1}$, with $c_0, \ldots, c_{\frac{i}{r}-1}$ in the first partition, $c_{\frac{i}{r}}, \ldots, c_{\frac{2i}{r}-1}$ in the second partition and so on. We denote as $C\colon \{0,1\}^i \rightarrow \{0,1\}^q$ the concatenation of outputs from all the Layer 1 PUFs. For a set $X \subseteq \{0,1\}^i$, we denote as $C[X]$ the set $\{C(y) \in \{0,1\}^q \mid y \in X\}$, i.e., all possible outputs from the Layer 1 PUFs on inputs from $X$. Thus, $C\left[\{0,1\}^i\right]$ is the range of $C$.

Suppose the set of queries the attacker issues to the black-box is $B \subseteq \{0,1\}^i$. The corresponding set of outputs is $C[B]$. Then, we have the following claim.

Claim 1. *If a PUF from composition* $[\mathbf{i}, \mathbf{m}, \mathbf{r}, \mathbf{s}]$ *has been fully characterized after queries from* $B \subseteq \{0,1\}^i$*, then* $C[B] = C\left[\{0,1\}^i\right]$.

We can prove the above claim by contradiction. If there exists some input $y \in \{0,1\}^i$ such that $C(y) \notin C[B]$, then given that the Layer 2 PUF is a random function, the attacker has at best a $1/2$ probability of guessing the output of the PUF $[\mathbf{i}, \mathbf{m}, \mathbf{r}, \mathbf{s}]$ on input $y$.

Claim 2. *There exists a PUF from composition* $[\mathbf{i}, \mathbf{m}, \mathbf{r}, \mathbf{s}]$ *for which* $\left|C\left[\{0,1\}^i\right]\right| \geq 2^q$*, where* $q = r\left(\left\lceil \frac{\frac{i}{r}-m}{m-s} \right\rceil + 1\right)$.

The proof for the above claim is by observing that if every Layer 1 PUF is xor, $\oplus$, of its input bits, then we have a sequence $c_0, c_1, \ldots, c_q$ of Layer 1 PUFs, where $q = r\left(\left\lceil \frac{\frac{i}{r}-m}{m-s} \right\rceil + 1\right)$, such that $c_j$ has an input that is not input to any $c_k$ where $k < j$. That is, every PUF as we go forward in that sequence has a "new" input. Thus, the concatenation of outputs of that sequence of Layer 1 PUFs is every bit string from $\{0,1\}^q$ when those Layer 1 PUFs are xor of their respective inputs.

*Consequence.* Claim 2 establishes a lower bound on the number of possible outputs from the Layer 1 PUFs. Claim 1 establishes that a lower bound on the attacker is that the number of black-box queries she issues must be at least the number of possible outputs from the Layer 1 PUFs. Together then, Claims 1 and 2 establish that a lower bound on the number of queries to the black-box that an attacker must issue to be confident that she has fully characterized $[\mathbf{i}, \mathbf{m}, \mathbf{r}, \mathbf{s}]$ is $2^q$, where $q = r\left(\left\lceil \frac{\frac{i}{r}-m}{m-s} \right\rceil + 1\right)$.

If $s = \Theta(m)$, $r = \Theta(1)$ and $m = O(i)$, then $q = \Theta(i)$. That is, under the assumption that the constituent PUFs are random functions, there exist PUFs that result from the composition whose strength is asymptotically bounded tightly by exactly the maximum possible CRP space, $2^i$. Example values for $s, r$ and $m$ that meet the sufficient condition for this to be achieved are $s = m - 1$, $r = 1$ and $m = i/16$.

We can compare the above lower-bound to the upper-bound to successfully attack a CPUF [4]. There, notwithstanding what the constituent PUFs are, an upper-bound number of queries to the black-box for a successful attack is $\frac{i}{m}2^m + 2^{\frac{i}{m}}$. Measured as bits, this strength is $O\left(\max\{m, \frac{i}{m}\}\right)$. This suggests that our broader admittance of ways to compose PUFs can be effective in yielding PUFs whose strength matches that of the maximum possible for a particular input size. The CPUF merely happens to be a weak member of the family.

## 5 MODELING ATTACKS

We now shift our focus to how our constituent PUFs are actually built. We start by considering attacks against PUFs that have been shown to be effective in prior work. In particular, we focus on a composite PUF whose constituent PUFs are ARB-PUFs. The state-of-the-art modeling attack against ARB-PUF is the logistic regression (LR) attack [14] that uses a linear additive model (LAM) to model the PUF. However, directly applying this attack to CPUF does not work as it does not capture multiple layers of PUFs in the architecture. Authors in [4] proposed a specific attack targeting composite PUFs, which they call a cryptanalysis attack (CA-ATK). We review CA-ATK [4], and present LR-CA-ATK on composite PUFs made up from ARB-PUFs that uses machine learning for its second phase. Note that both CA-ATK and LR-CA-ATK focus on modeling CPUF instances, which corresponds to $s = 0$ in the family we discuss.

### 5.1 Overview of CA-ATK

CA-ATK is split into two phases and in this section we focus on constructing the model for a CPUF $P$ $[\mathbf{i} = nm, \mathbf{m} = m, \mathbf{r} = n, \mathbf{s} = 0]$.
**Phase 1.** In the first phase, for each Layer 1 PUF $p_i$, the attacker partitions the challenge space $C_i = \{0,1\}^m$ of $p_i$ into two sets $S_{i,0}$ and $S_{i,1}$. Let us assume that $\mathbf{a}$ and $\mathbf{b}$ are two challenges in $C_i$. If $\mathbf{a} \in S_{i,0}$ and $\mathbf{b} \in S_{i,1}$ then $p_i(\mathbf{a}) \neq p_i(\mathbf{b})$. This means that the two challenges produce different outputs for $p_i$. Similarly, if $\mathbf{a}, \mathbf{b} \in S_{i,j}, j \in \{0,1\}$, then $p_i(\mathbf{a}) = p_i(\mathbf{b})$ with high likelihood, which means the two challenges are likely to produce the same output for $p_i$. To construct the sets $S_{i,0}$ and $S_{i,1}$, one selects two $m$-bit challenges from $C_i$, and extends them to $mn$-bit challenges $\mathbf{x}$ and $\mathbf{y}$ by simply keeping the extended bits the same in both extended versions. If $P(\mathbf{x}) \neq P(\mathbf{y})$, then the attacker can be sure that the corresponding $m$-bit challenges are not in the same set. If $P(\mathbf{x}) = P(\mathbf{y})$, then the attacker can not be sure that they are in the same set. However, one can repeat this procedure multiple times by changing the extended bits.
**Phase 2.** In this phase, the attacker uses the information from the first phase to enumerate the challenge space of the Layer 2 PUF. Algorithm 1 shows the steps in the second phase. Notice that the collected information in $S_{i,0}$ and $S_{i,1}$ is used to construct an input challenge to the composed PUF. It does this by selecting an $n$-bit

Z. Wu, H. Patel, M. Sachdev, and M. Tripunitara, "Strengthening PUFs using Composition," in proceedings of IEEE/ACM International Conference On Computer Aided Design (ICCAD), Nov. 2019, pp. 1–8. doi: 10.1109/ICCAD45719.2019.8942176.



**Algorithm 1:** Class-Construction($S$, $P$)

**Input** : $S = \{S_{i,0}, S_{i,1} | 0 \le i < n\}$ and CPUF $P$
**Output**: Set of special CRPs $Y$ and the response vector $\mathbf{y} \in \{0,1\}^n$

$Y = \emptyset$;
$\mathbf{y} = \mathbf{0}$;
**for** $\mathbf{u} = (u_0, u_1, ..., u_{n-1}) \in \{0,1\}^n$ **do**
  $\mathbf{c} = (\mathbf{c}_0, \mathbf{c}_1, ..., \mathbf{c}_{n-1}), \mathbf{c}_i \in S_{i,u_i}$;
  $y_{\mathbf{u}} = P(\mathbf{c})$;
  $Y = Y \cup \{(\mathbf{c}, y_{\mathbf{u}})\}$;

**Algorithm 2:** Enhanced-Class-Construction($S$, $P$, $D$)

**Input** : $S = \{S_{i,0}, S_{i,1} | 0 \le i < n\}$, CPUF $P$ and $D \subseteq C = \{0,1\}^{nm}$ is a set of $N$ challenges of $P$.
**Output**: A LAM for $P_n$

$E = \emptyset$;
**for** $\mathbf{c} \in D$ **do**
  $\mathbf{u} = (u_0, u_1, ..., u_{n-1})$, where $\mathbf{c}_i \in S_{i,u_i}$;
  $E = E \cup \{(\mathbf{u}, P(\mathbf{c}))\}$;
Compute model $P_n$ with LR, given CRP set $E$;

string $\mathbf{u}$, and replacing one of the sub-strings $u_i$ with an $m$-bit string from the first phase. The resulting string $\mathbf{c}$ is applied to the composed PUF $P$, and the challenge response pair is saved. Notice that this approach requires enumerating the $n$-bit challenge space.

## 6 LR-CA-ATK: ENHANCED ATTACK

We propose an enhanced attack on ARB-PUF based CPUF which exploits the property of outputs of Layer 1 PUFs and LAM for ARB-PUF. In phase 1 of CA-ATK, we make a guess about the Layer 1 PUFs by assigning their challenges to $S_{i,0}$ or $S_{i,1}$. We show that the guess can be characterized by a vector $\mathbf{x} \in \{0,1\}^n$ and this can be exploited to construct the LR-CA-ATK. In this section, we focus on constructing the model for a PUF [$\mathbf{i} = nm, \mathbf{m} = m, \mathbf{r} = n, \mathbf{s} = 0$].

We denote the attacker's guess for the $i$-th Layer 1 PUF as $\tilde{p_i}$, which is constructed in phase 1. Thus, $\tilde{p_i}(\mathbf{a}) = j$, for $\mathbf{a} \in S_{i,j}$. The number of possible guesses an attacker makes about the output of the Layer 1 PUFs is large, however, we show that this can be fully characterized by an $n$-bit string. Specifically, we show that this applies to every Layer 1 PUF $p_i$ in Theorem 1.

Theorem 1. *For all $0 \le i < n$, $\mathbf{c} \in \{0,1\}^m$, $\tilde{p_i}(\mathbf{c}) = p_i(\mathbf{c}) \oplus x_i$, where $x_i \in \{0,1\}$ and $\oplus$ is binary exclusive-or.*

Proof. We choose an arbitrary challenge of $p_i$: $\mathbf{d} \in \{0,1\}^m$. If $p_i(\mathbf{c}) = p_i(\mathbf{d})$, then $\mathbf{c}$ and $\mathbf{d}$ belong to the same set; thus, $\tilde{p_i}(\mathbf{c}) = \tilde{p_i}(\mathbf{d})$. If $p_i(\mathbf{c}) \neq p_i(\mathbf{d})$, then $\mathbf{c}$ and $\mathbf{d}$ belong to different sets; thus, $\tilde{p_i}(\mathbf{c}) = 1 \oplus \tilde{p_i}(\mathbf{d})$. In both cases, $\tilde{p_i}(\mathbf{c}) = p_i(\mathbf{c}) \oplus \tilde{p_i}(\mathbf{d}) \oplus p_i(\mathbf{d})$ holds. □

According to Theorem 1, the attacker's guess of $p_i$ is determined by a secret bit $x_i = \tilde{p_i}(\mathbf{d}) \oplus p_i(\mathbf{d})$. Another fact we utilize is the LAM used for attacking ARB-PUF. In [13], the ARB-PUF $P$ with $n$-bit challenge can be modeled as

$$P(\mathbf{c}) = sgn(\mathbf{w}\phi) = sgn(\mathbf{w}F(\mathbf{c})),$$

where $\mathbf{w} \in \mathbb{R}^{n+1}$ and $\phi \in \{-1,1\}^{n+1}$ is defined as:

$$\phi_i = F(\mathbf{c})(i) = \begin{cases} (-1)^{c_i}\phi_{i+1} = \prod_{j=i}^{n}(-1)^{c_j}, & \text{if } 0 \le i \le n-1 \\ 1, & \text{if } i = n \end{cases}. \quad (1)$$

We then show in Theorem 2 that if the challenge of an $n$-bit ARB-PUF $P$ is XOR'ed with a secret key $\mathbf{x} \in \{0,1\}^n$, it can still be modeled with a LAM. The theorem shows that an ARB-PUF with XOR operation between a secret vector and its challenge is equivalent to another instance of ARB-PUF. Also, this theorem allows us to exploit the property of the attacker's guess in Theorem 1.

Theorem 2. *$P(\mathbf{c} \oplus \mathbf{x}) = sgn(\mathbf{w}F(\mathbf{c} \oplus \mathbf{x})) = sgn(\mathbf{v}F(\mathbf{c}))$, where $\mathbf{w}, \mathbf{v} \in \mathbb{R}^{n+1}$ and $P$ is an $n$-bit ARB-PUF.*

If we choose $v_i = w_i \prod_{j=i}^{n}(-1)^{x_j}$ for $i < n$ and $v_n = w_n$, then Theorem 2 holds.

Now let us combine Theorem 1 and Theorem 2 to illustrate the weakness of CPUF. According to Theorem 1, the guess made in phase 1 of the CA-ATK about the output Layer 1 PUFs is actually not *far* from the actual output of these PUFs. If one knows about the secret string $\mathbf{x}$, this person could derive the actual output of the Layer 1 PUFs. And Theorem 2 shows that if the Layer 2 PUF is an ARB-PUF, this ARB-PUF is equivalent to another ARB-PUF, whose CRP space is defined by the guessed output and the original responses. These observations imply that an attacker does not need to explicitly obtain the secret string $\mathbf{x}$ to build the model for the Layer 2 ARB-PUF. Previous work has shown that building models for ARB-PUF with LR can be done efficiently [13], which we exploit here. We formalize these observations in Theorem 3.

Theorem 3. *Let $P$ be a [$\mathbf{i} = nm, \mathbf{m} = m, \mathbf{r} = n, \mathbf{s} = 0$]. $\mathbf{c} \in \{0,1\}^{nm}$ is a challenge of $P$, $\mathbf{r} = (p_0(\mathbf{c_0}), p_1(\mathbf{c}_1), ..., p_{n-1}(\mathbf{c}_{n-1}))$ is the response vector of Layer 1 PUFs, $\tilde{\mathbf{r}} = (\tilde{p_0}(\mathbf{c_0}), \tilde{p_1}(\mathbf{c}_1), ..., \tilde{p}_{n-1}(\mathbf{c}_{n-1}))$ is the response vector constructed in phase 1 of the CA-ATK and $\mathbf{x}$ is a vector in $\{0,1\}^n$ that determines the guess, then*

$$P(\mathbf{c}) = p_n(\mathbf{r}) = p_n(\tilde{\mathbf{r}} \oplus \mathbf{x}) = sgn(\mathbf{v}F(\tilde{\mathbf{r}})),$$

*where $p_n$ is the Layer 2 PUF.*

Algorithm 2 shows the procedure of using Theorem 3 to model the Layer 2 ARB-PUF. The attacker first randomly chooses a set $D$ of $nm$-bit challenges. For each challenge in $D$, the attacker applies the reverse transformation as is done in Algorithm 1 to get a set of challenges of $n$-bit. This reversed transformation replaces every $m$-bit component with a single bit, based on whether the $m$-bit component is in $S_{\cdot,0}$ or $S_{\cdot,1}$. These challenges, combined with their corresponding responses, are fed into the LR attack to train a model for the CPUF. Compared to Algorithm 1, Algorithm 2 does not require the enumeration of all strings in $\{0,1\}^n$. The size of CRP set required is $N = |D|$, which can be much less than $2^n$.

In summary, for an $n$-bit $P \circ P$, the attacker needs to enumerate $2^n$ CRPs to build the model. This is impractical for large values of $n$. Thus, an attacker cannot efficiently attack $P \circ P$ with this technique. Furthermore, an attacker could attack ARB-PUF based CPUF more efficiently yet $P \circ P$ is not subject to this.

Z. Wu, H. Patel, M. Sachdev, and M. Tripunitara, "Strengthening PUFs using Composition," in proceedings of IEEE/ACM International Conference On Computer Aided Design (ICCAD), Nov. 2019, pp. 1–8. doi: 10.1109/ICCAD45719.2019.8942176.



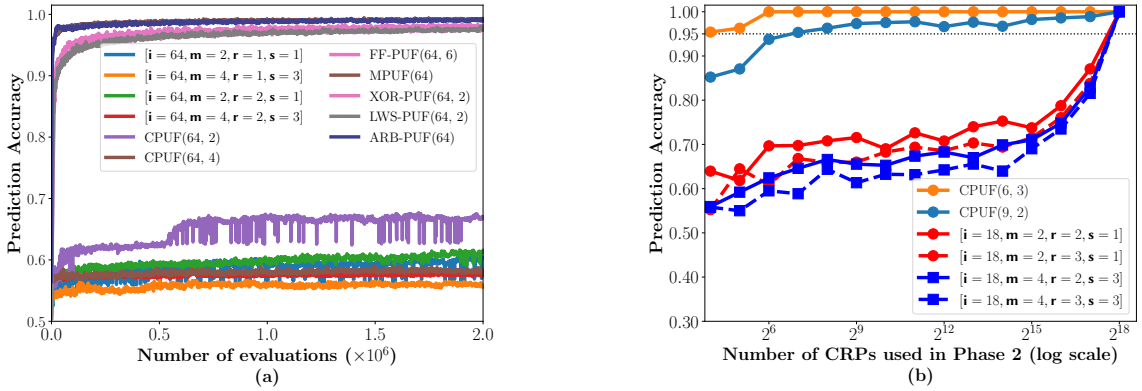



**Figure 3: (a) $P \circ P$ versus state-of-the-art PUFs using ES modeling attack, and (b) $P \circ P$ against CPUF while varying the number of training CRPs for Phase 2 of LR-CA-ATK.**

## 7 EMPIRICAL EVALUATION

We separate our evaluation into three parts. The first part compares $P \circ P$ with state-of-the-art PUFs including CPUF composed using ARB-PUFs using an evolutionary strategy (ES) attack [13]. Using insights from the first part, we proceed to the second part. The second part shows that LR-CA-ATK successfully models CPUF, but it is unable to model $P \circ P$ with various mapping functions. Using these results, we provide insights into properties that make a mapping function resilient to LR-CA-ATK. We accomplish this by carefully selecting a few different mapping functions, and comparing them. The third part evaluates statistical performance metrics of $P \circ P$ including training times for CA-ATK and LR-CA-ATK on CPUF. We also make some observations on the incurred hardware.

### 7.1 Experimental Setup

We perform our empirical evaluation using a simulation framework built on Tensorflow [6]. Our simulation framework is publicly available for download [15]. For LR, we implement RProp [12], and for ES, we use the open-source implementation provided by PyBrain [11]. We use the meta-parameters for ES from [14] that have resulted in successful attacks on a variety of PUF architectures. Without loss of generality, our results use PUF instances with an 18-bit challenge. The 18-bit challenge allows us to explore different mapping functions while the attacks can still be practically conducted.

### 7.2 Results

We begin by showing that CPUF is vulnerable to LR-CA-ATK, but $P \circ P$ remains resilient. This establishes that $P \circ P$ with varying mapping functions results in a strengthened architecture with respect to LR-CA-ATK. We use this result to discover properties of $P \circ P$ that contribute to its resilience. In particular, we investigate varying the input size of the Layer 1 PUFs, and the effect of mapping functions on the amount of sharing across Layer 1 PUFs.

*7.2.1 Comparison against state-of-the-art PUF architectures.* We use the notation described in Table 1 to refer to PUFs, and their configurations. We compare four $P \circ P$ configurations: $[\mathbf{i} = 64, \mathbf{m} = 2, \mathbf{r} = 1, \mathbf{s} = 1]$, $[\mathbf{i} = 64, \mathbf{m} = 4, \mathbf{r} = 1, \mathbf{s} = 3]$, $[\mathbf{i} = 64, \mathbf{m} = 2, \mathbf{r} = 2, \mathbf{s} = 1]$ and $[\mathbf{i} = 64, \mathbf{m} = 4, \mathbf{r} = 2, \mathbf{s} = 3]$ against two CPUF architectures CPUF(64, 2) and CPUF(64, 4), FF-PUF(64, 6), MPUF(64, 1), XOR-PUF(64, 2), ARB-PUF, LWS-PUF(64, 2) as shown in Figure 3a. We use ES to attack the aforementioned PUFs as ES only requires a parametric model. For all the PUFs in Figure 3a, the training set and test set include 50,000 and 10,000 randomly generated CRPs, respectively. We observe that $[\mathbf{i} = 64, \mathbf{m} = 2, \mathbf{r} = 1, \mathbf{s} = 1]$, $[\mathbf{i} = 64, \mathbf{m} = 4, \mathbf{r} = 1, \mathbf{s} = 3]$, $[\mathbf{i} = 64, \mathbf{m} = 2, \mathbf{r} = 2, \mathbf{s} = 1]$ and $[\mathbf{i} = 64, \mathbf{m} = 4, \mathbf{r} = 2, \mathbf{s} = 3]$ sustain a prediction accuracy of 60.48%, 55.87%, 61.57% and 57.70% even after performing $2 \times 10^6$ evaluations. Other PUFs have the following prediction accuracy: FF-PUF with 6 loops has 97.95%, XOR-PUF has 97.6%, LWS-PUF has 97.47%, ARB-PUF has 99.14% and MPUF has 99.3%. Note that all alternative PUF architectures exhibit an increase in prediction accuracy to more than 97%. This shows that a larger effort is required to attack $P \circ P$ and CPUF than the alternatives with ES. For instances of $P \circ P$, the architecture with the same number of partitions, the ones with more Layer 1 PUF stages show lower prediction accuracy. The difference is within 5%. For $P \circ P$ instances with the same Layer 1 PUF stages, the ones with fewer partition numbers show lower prediction accuracy. The difference is within 2%. These experiments confirm that $P \circ P$ provides additional resistance to ML modeling attacks with ES when compared against other PUFs.

*7.2.2 Comparing Resilience of CPUF against $P \circ P$ using LR-CA-ATK.* Figure 3b shows the prediction accuracy of LR-CA-ATK when

**Table 1: The notation of different PUF architectures.**

| Notation | Architecture |
|---|---|
| FF-PUF($n$, $l$) | $n$-bit FF-PUF with $l$ loops |
| XOR-PUF($n$, $c$) | $n$-bit XOR-PUF with $c$ chains |
| LWS-PUF($n$, $c$) | $n$-bit LWS-PUF with $c$ chains |
| MPUF($n$) | $n$-bit MPUF. |
| ARB-PUF($n$) | $n$-bit ARB-PUF. |

Z. Wu, H. Patel, M. Sachdev, and M. Tripunitara, "Strengthening PUFs using Composition," in proceedings of IEEE/ACM International Conference On Computer Aided Design (ICCAD), Nov. 2019, pp. 1–8. doi: 10.1109/ICCAD45719.2019.8942176.



applied to CPUF and $P \circ P$ with a varying number of CRPs. The prediction accuracy is a metric to describe a PUF's resistance to modeling attacks. A high prediction accuracy signifies less effort is required to model the PUF compared to one with lower prediction accuracy. Recall, there are two phases in LR-CA-ATK. The experiment in Figure 3b varies the number of CRPs used in Phase 2 of LR-CA-ATK. When we evaluate the model, we exclude the CRPs used in Phase 1 and Phase 2. The reason for this is that any CRP used for training the model is stored by the attacker since the PUF is actually exercised. We consider a PUF to be modeled when the prediction accuracy is around 95%. Figure 3b shows that CPUF is modeled with more than 95% prediction accuracy when the number of the training CRPs is $2^7$. However, $2^{13}$ training CRPs, LR-CA-ATK cannot model our proposed architectures with 95% prediction accuracy. Even if we use half of the total CRP space, our results show that we cannot model $P \circ P$ with more than 95% prediction accuracy. Since increasing the number of training CRPs does not help us model $P \circ P$, we choose an in-between value of $2^{13}$ as the number of training CRPs in Phase 2 for the remaining experiments in Section 7.2.3.

*7.2.3 Evaluating properties of mapping functions.* We evaluate two properties of mapping functions: 1) the resulting partition size, and 2) the number of partitions from mapping functions. We assess the security with the *normalized prediction accuracy*, which is the prediction accuracy divided by the number of CRPs used in Phase 1 and Phase 2. This metric measures the prediction accuracy achieved per CRP used. For example, a prediction accuracy of 60% with $2^{13}$ training CRPs in Phase 2 and 1024 CRPs in Phase 1 results in a normalized prediction accuracy of $6.5 \times 10^{-5}$. A PUF with a higher normalized prediction accuracy means the CRP utilization is high rendering the PUF less secure.

**Partition size as a result of mapping functions.** We investigate the effect of partition size resulting from the mapping function. In Figure 4a, we evaluate a class of mapping functions, which partition Layer 1 PUFs into two partitions ($[\mathbf{i} = 18, \mathbf{m} = m, \mathbf{r} = 2, \mathbf{s} = 2]$). Here, the partition sizes are not necessarily the same. The lowest normalized prediction accuracy occurs when the size of the first partition is small or large compared to the other one. The normalized prediction accuracy peaks at the point where the partition sizes are equal and such a mapping function is less desirable as it offers reduced strength. This is because, if one of the partitions induced by the mapping function has a large size, the attacker must enumerate the larger partition in Phase 1 of the attack. Thus, a mapping function where a partition for Layer 1 PUFs has a large size is beneficial.

**Number of partitions as a result of mapping functions.** We investigate the effect of number of partitions resulting from the mapping function. We select mapping functions that divide Layer 1 PUFs into equally sized groups. Figure 4b shows that as the number of partitions increase, the normalized prediction accuracy increases and an attacker can model the PUF effectively. This means that a mapping function that lowers the number of partitions of Layer 1 PUFs is better.

**Suggestions for composition.** Our study reveals that a good mapping function has the following. 1) A large partition size such that an attacker must enumerate the larger partition. 2) A small number of partitions such that an attacker can achieve a lower normalized prediction accuracy. Using these as guides, we select the configurations of 64-bit $P \circ P$. LR-CA-ATK cannot model these configurations due to the large CRP space. Hence, we use ES to show their resilience to the attack as presented in section 7.2.1.

**Table 2: LR-CA-ATK and CA-ATK on CPUF. Prediction rates and the training time are averaged over 5 trials.**

| CPUF Types | Pred. Acc. LR-CA-ATK | Pred. Acc. CA-ATK | Train. Time LR-CA-ATK | Train. Time CA-ATK |
|---|---|---|---|---|
| CPUF(20, 4) | **99.37%** | 99.92% | **31.38s** | 162.91s |
| CPUF(64, 4) | **99.62%** | - | **33.94s** | - |

*7.2.4 Training time versus accuracy.* Table 2 compares the training time and accuracy of LR-CA-ATK against CA-ATK. Note that it is impractical to apply CA-ATK to CPUF(64, 4) as its Phase 2 requires the enumeration over a space of $2^{64}$ binary strings. The result clearly shows that LR-CA-ATK succeeds in modeling both CPUFs, but CA-ATK fails to complete for CPUF(64, 4). In addition, LR-CA-ATK only takes 20% of the time CA-ATK takes for CPUF(20, 4). Table 3 shows the training time and the best achieved prediction rate for 50,000 CRPs. We observe that $[\mathbf{i} = 64, \mathbf{m} = 2, \mathbf{r} = 1, \mathbf{s} = 1]$ has a higher prediction accuracy by ~4.6% over $[\mathbf{i} = 64, \mathbf{m} = 4, \mathbf{r} = 1, \mathbf{s} = 3]$, but the training time for $[\mathbf{i} = 64, \mathbf{m} = 2, \mathbf{r} = 1, \mathbf{s} = 1]$ is larger due to the fact that the model with 4-stage Layer 1 PUFs has higher complexity. This result suggests that $[\mathbf{i} = 64, \mathbf{m} = 2, \mathbf{r} = 1, \mathbf{s} = 1]$ provides just as much resilience as $[\mathbf{i} = 64, \mathbf{m} = 4, \mathbf{r} = 1, \mathbf{s} = 3]$. Also, both $P \circ P$ configurations outperform FF-PUFs in terms of their prediction accuracy.

**Table 3: Prediction accuracy for the best of 40 trials for $P \circ P$ and other PUFs. The training time is the averaged.**

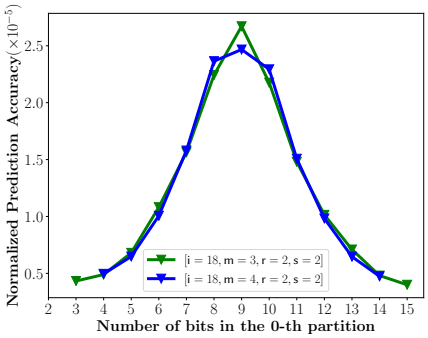
| Architecture | Pred. Acc. Best Run | Training Time |
|---|---|---|
| $[\mathbf{i} = 64, \mathbf{m} = 2, \mathbf{r} = 1, \mathbf{s} = 1]$ | 60.48% | 1:57 hrs |
| $[\mathbf{i} = 64, \mathbf{m} = 4, \mathbf{r} = 1, \mathbf{s} = 3]$ | 55.87% | 2:29 hrs |
| $[\mathbf{i} = 64, \mathbf{m} = 2, \mathbf{r} = 2, \mathbf{s} = 1]$ | 61.57% | 1:53 hrs |
| $[\mathbf{i} = 64, \mathbf{m} = 4, \mathbf{r} = 2, \mathbf{s} = 3]$ | 57.70% | 2:31 hrs |
| CPUF(64, 2) | 66.86% | 1:35 hrs |
| CPUF(64, 4) | 58.16% | 1:55 hrs |
| FF-PUF(64, 6) | 97.95% | 5:59 hrs |
| XOR-PUF(64, 2) | 97.60% | 0:20 hrs |
| LWS-PUF(64, 2) | 97.47% | 0:21 hrs |
| MPUF(64) | 99.30% | 0:12 hrs |
| ARB-PUF(64) | 99.14% | 0:31 hrs |

## 7.3 Performance Metrics

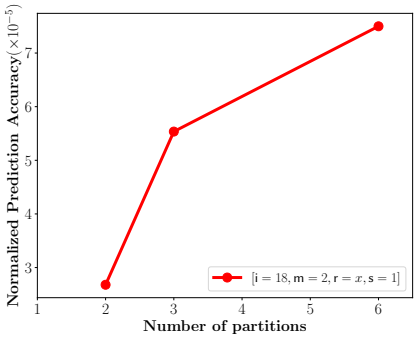
Table 4 presents the uniformity and uniqueness performance metrics. Our results show that $P \circ P$ with the different mapping functions offer good uniqueness traits. However, for $P \circ P$s with $\mathbf{r} = 2$, the uniformity is biased. Note that we do not include the effects of noise, bias, and reliability of the construction in this work.

Z. Wu, H. Patel, M. Sachdev, and M. Tripunitara, "Strengthening PUFs using Composition," in proceedings of IEEE/ACM International Conference On Computer Aided Design (ICCAD), Nov. 2019, pp. 1–8. doi: 10.1109/ICCAD45719.2019.8942176.






**(a) $P \circ P$ as a function of number of bits of the first partition (overlapped bits between two neighbouring L1 PUFs: 2).**




**(b) $P \circ P$ as a function of number of partitions.**

**Figure 4: Normalized prediction accuracy given different properties of mapping functions.**

**Table 4: Uniqueness and uniformity for $P \circ P$.**

| PUF Architectures | Uniqueness | Uniformity |
|---|---|---|
| [**i** = 64, **m** = 4, **r** = 1, **s** = 3] | 50.4% | 47.6% |
| [**i** = 64, **m** = 4, **r** = 2, **s** = 3] | 50.5% | 61.7% |
| [**i** = 64, **m** = 2, **r** = 1, **s** = 1] | 49.5% | 49.5% |
| [**i** = 64, **m** = 2, **r** = 2, **s** = 1] | 50.4% | 63.6% |

## 7.4 Hardware Cost

In Table 5, we compare the hardware cost of $P \circ P$ and CPUF. [**i** = 64, **m** = 2, **r** = 1, **s** = 1] and [**i** = 64, **m** = 4, **r** = 1, **s** = 3] are the $P \circ P$ instances that take 64-bit challenge. [**i** = 64, **m** = 4, **r** = 1, **s** = 3] is slightly better than [**i** = 64, **m** = 2, **r** = 1, **s** = 1] according to our evaluations with the ES attack. This enhanced security comes at the price of 66.7% larger number of stages. [**i** = 64, **m** = 4, **r** = 1, **s** = 3] and CPUF(64, 4) have the same hardware cost and CPUF(64, 2) can take 128-bit challenge. However, we already showed that CPUF(64, 4) can be modeled using LR-CA-ATK with far fewer CRPs than the 128-bit challenge space.

**Table 5: The hardware cost of different PUF compositions.**

| PUF Architectures | ARB-PUF(2) | ARB-PUF(4) | ARB-PUF(64) | Number of Stages |
|---|---|---|---|---|
| [**i** = 64, **m** = 2, **r** = 1, **s** = 1] | 64 | – | 1 | 192 |
| [**i** = 64, **m** = 4, **r** = 1, **s** = 3] | – | 64 | 1 | 320 |
| CPUF(64, 2) | 64 | – | 1 | 192 |
| CPUF(64, 4) | – | 64 | 1 | 320 |

## 8 CONCLUSION

We revisited the idea of strengthening PUFs by constructing PUFs that are compositions of other PUFs. We proposed a general model for composition, and considered a particular family in that model that admits new kinds of compositions, and also captures prior constructions. We established analytically that even within this restricted family of constructions, there can exist PUFs whose strength, asymptotically, is the maximum possible for a particular input-size. We then revisited state-of-the-art attacks on PUFs, and proposed an enhancement to the prior attack on such compositions. Via our empirical assessments, we confirmed that constructing PUFs by composition is indeed a promising approach to realizing strong PUFs. Our general model, and the manner in which PUFs are realized in practice, suggest a rich area of future work.

## REFERENCES

[1] A. Vijayakumar et al. 2016. Machine learning resistant strong PUF: Possible or a pipe dream?. In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. 19–24.
[2] C. Herder et al. 2014. Physical Unclonable Functions and Applications: A Tutorial. *Proc. IEEE* 102, 8 (Aug 2014), 1126–1141.
[3] D. P. Sahoo et al. 2014. Composite PUF: A new design paradigm for Physically Unclonable Functions on FPGA. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*. 50–55.
[4] D. P. Sahoo et al. 2015. A Case of Lightweight PUF Constructions: Cryptanalysis and Machine Learning Attacks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* 34, 8 (Aug 2015), 1334–1343.
[5] G. E. Suh et al. 2007. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In *ACM/IEEE Design Automation Conference (DAC)*. 9–14.
[6] M. Abadi et al. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/
[7] M. Majzoobi et al. 2008. Lightweight secure PUFs. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 670–673.
[8] Q. Ma et al. 2018. A machine learning attack resistant multi-PUF design on FPGA. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*. 97–104.
[9] S. K. Mathew et al. 2014. A 0.19pJ/b PVT-variation-tolerant hybrid physically unclonable function circuit for 100% stable secure key generation in 22nm CMOS. In *IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. 278–279.
[10] T. H. Cormen et. al. 2009. *Introduction to Algorithms* (3rd ed.). MIT Press.
[11] T. Schaul et al. 2010. PyBrain. *Journal of Machine Learning Research* 11, Feb (2010), 743–746.
[12] U. Rührmair et al. 2010. Retrieved "May 3, 2018" from http://www.pcp.in.tum.de/code/lr.zip
[13] U. Rührmair et al. 2010. Modeling Attacks on Physical Unclonable Functions. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*. 237–249.
[14] U. Rührmair et al. 2013. PUF Modeling Attacks on Simulated and Silicon Data. *IEEE Transactions on Information Forensics and Security (TIFS)* 8, 11 (2013), 1876–1891.
[15] Z. Wu et al. 2019. https://git.uwaterloo.ca/caesr-pub/pop-iccad-2019
[16] C. Hou. 2017. A smart design paradigm for smart chips. In *IEEE International Solid-State Circuits Conference (ISSCC)*. 8–13.
[17] D. Mukhopadhyay. 2016. PUFs as Promising Tools for Security in Internet of Things. *IEEE Design Test* 33, 3 (June 2016), 103–115.